# Ethernet Protocol Stack Chip CH392

Datasheet

Version: 2.2

https://wch-ic.com

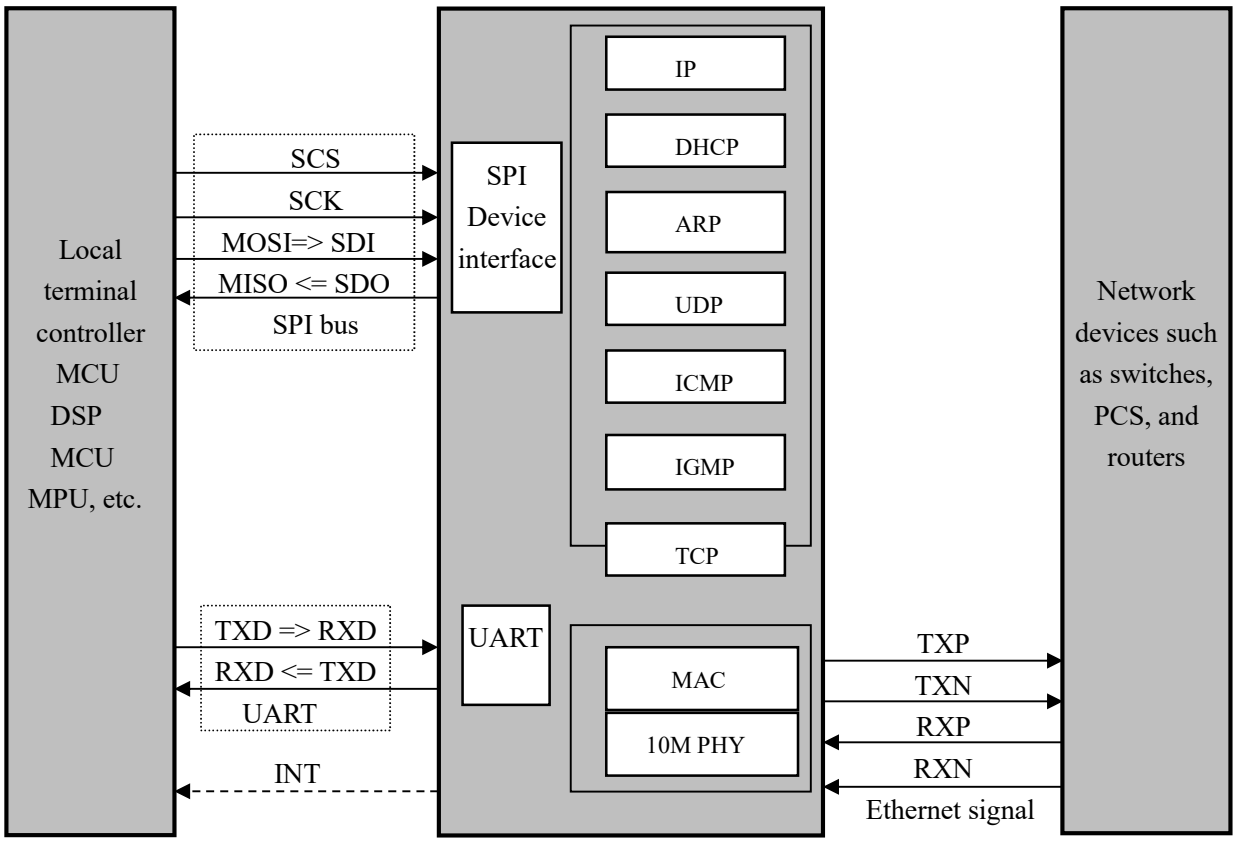
## 1. Overview

CH392 is an Ethernet protocol stack management chip, which is used for Ethernet communication in single-chip microcomputer system.

The CH392 chip comes with a 10M Ethernet Media Access Control (MAC) and physical layer (PHY), fully compatible with IEEE802.3, and built-in Ethernet protocol stack firmware such as IP, DHCP, ARP, ICMP, IGMP, UDP, and TCP. The single chip microcomputer system can conveniently communicate through CH392 chip.

The CH392 supports 2 communication interfaces: SPI interface or UART. The controller such as single-chip microcomputer/DSP/MCU/MPU can control CH392 for Ethernet communication through any of the above communication interfaces.

The following is the application block diagram of CH392.

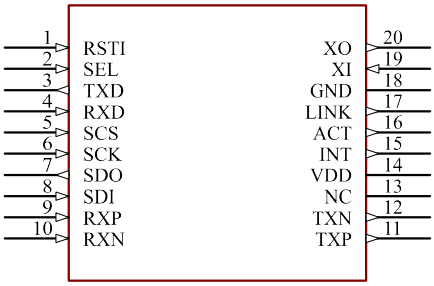



## 2. Features

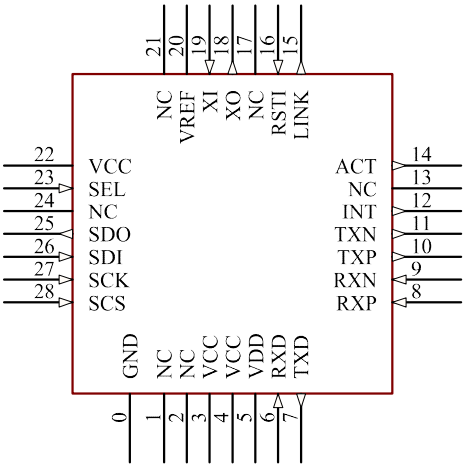
- Internal Ethernet MAC and PHY.
- Support 10M, full-duplex/half-duplex adaptive, compatible with 802.3 protocol.
- Support automatic conversion of MDI/MDIX lines.
- Built-in TCP/IP protocol stack, and support IPv4, DHCP, ARP, ICMP, IGMP, UDP and TCP protocols.
- Up to 8 independent Socket pairs (CH392T) can be used to send and receive data simultaneously.

- Provide two interfaces: SPI and serial port.
- Support MACRAW mode (CH392T).
- Available in TSSOP20 and QFN28 packages.

## 3. Packages




CH392T(TSSOP20)




CH392F(QFN28)

| Package Form | Body Size | Pin Pitch | | Package Description | Order Model |
|---|---|---|---|---|---|
| TSSOP20 | 4.4*6.5mm | 0.65mm | 25.6mil | Thin Shrink Small Outline Package | CH392T |
| QFN28 | 4.0*4.0mm | 0.40mm | 15.7mil | Quad Flat No-Lead Package | CH392F |

*Note: CH392T is an upgrade of CH392F. The CH392T is recommended for the new design.*

# 4. Pin Definitions

1. CH392T

| CH392T Pin No. | Pin name | Type | Pin description |
|---|---|---|---|
| 1 | RSTI | Input | External reset input, active low |
| 2 | SEL | Input | Communication interface select pin, built-in pull-up, high level selects serial port, low level selects SPI interface |
| 3 | TXD | Output | UART data output |
| 4 | RXD | Input | UART data input, built-in pull-up resistor |
| 5 | SCS | Input | SPI chip selection input |
| 6 | SCK | Input | SPI clock input |
| 7 | SDO | Output | SPI data output |
| 8 | SDI | Input | SPI data input |
| 9 | RXP | Ethernet signal | Ethernet RXP signal |
| 10 | RXN | Ethernet signal | Ethernet RXN signal |
| 11 | TXP | Ethernet signal | Ethernet TXP signal |
| 12 | TXN | Ethernet signal | Ethernet TXN signal |
| 13 | NC | NC | Reserved pins, suspended |
| 14 | $V_{DD}$ | Power | 3.3V supply voltage input |
| 15 | INT | Output | Interrupt signal output, active low |
| 16 | ACT | Output | Ethernet connection communication indicator driver pin |
| 17 | LINK | Output | PHY connection indicator pin, active low |
| 18 | GND | Power | Ground |
| 19 | XI | Input | The input end of the crystal oscillation |
| 20 | XO | Output | The input end of the crystal oscillation |

*Note: The CH392T requires an external 8M crystal before version 0x18, and an external 8M or 16MHz crystal is available for version 0x18 and later.*

2、CH392F

| CH392F Pin No. | Pin name | Type | Pin description |
|---|---|---|---|
| 0 | GND | Power | Ground |
| 3,4,22 | $V_{CC}$ | Power | Internal power supply, an external 0.1uF decoupling capacitor is required |
| 5 | $V_{DD}$ | Power | 3.3V power supply voltage input, external 2.2uF decoupling capacitor is required |
| 6 | RXD | Input | UART data input, built-in pull-up resistor |
| 7 | TXD | Output | UART data output, built-in pull-up resistor |
| 8 | RXP | Ethernet signal | Ethernet RXP signal |
| 9 | RXN | Ethernet signal | Ethernet RXN signal |
| 10 | TXP | Ethernet signal | Ethernet TXP signal |
| 11 | TXN | Ethernet signal | Ethernet TXN signal |
| 12 | INT | Output | Interrupt signal output, active low |
| 14 | ACT | Output | Ethernet connection communication indicator driver pin |
| 15 | LINK | Output | PHY connection indicator pin, active low |
| 16 | RSTI | Input | External reset input, active low |
| 18 | XO | Output | The reverse phase output of the crystal oscillation requires an external 32MHz crystal |
| 19 | XI | Input | The input end of the crystal oscillation requires an external 32MHz crystal |
| 20 | $V_{REF}$ | Power | The internal analog circuit power node needs an external 1uF decoupling capacitor |
| 23 | SEL | Input | Communication interface select pin, built-in pull-up, high level select serial port, low level select SPI |
| 25 | SDO | Output | SPI data output |
| 26 | SDI | Input | SPI data input |
| 27 | SCK | Input | SPI clock input |
| 28 | SCS | Input | SPI chip selection input |
| 1,2,13,17,21,24 | NC | NC | Reserved pins, suspended |

# 5. Commands

In this datasheet, the suffix B is a binary number, the suffix H is a hexadecimal number, otherwise it is a decimal number.

Little-Endian two-word data (total 32 bits) refers to the lowest byte (bit 7 to bit 0), followed by the lower byte (bit 15 to bit 8), then the higher byte (bit 23 to bit 16), and finally the highest byte (bit 31 to bit 24).

A data stream is a data block composed of several consecutive bytes. The total length of a data block ranges from 0 to 4096.

The numbers in parentheses of the input data and the output data in the following table are the number of bytes of the parameters. Without parentheses, the default is 1 byte.

The MCU referred to in this manual is basically applicable to DSP or MCU/MPU/SCM, etc.

Socket Pair contains the source IP address, source port, destination IP address, and destination port of the quadruple, can uniquely identify the two connected parties on the Internet. This manual is referred to as Socket. The CH392 can provide multiple sockets internally at the same time, and their index values are 0,1,2, 3 in sequence..... The CH392T provides eight sockets, and the CH392F provides four sockets.

The high and low bytes of IP and MAC addresses agreed in this manual may differ from some documents, for convenience only:

For example, if the IP address is 192.168.1.2, 192 is the lowest byte and 2 is the highest byte. This article refers to (IP) as low byte first.

For example, the MAC address is 00.01.02.03.04.05, where 00 is the lowest byte and 05 is the highest byte. This article refers to the (MAC) low byte before.

In this datasheet, the byte order of all commands that contain the input or output of IP addresses is the lowest byte before the IP address.

In this datasheet, the byte order of all commands that contain the input or output of MAC addresses is the MAC lower byte first.

| Code | Command name CMD_ | Input data | Output data | Command application |
|---|---|---|---|---|
| 01H | GET_IC_VER | | Version | Get the chip and firmware versions |
| 02H | SET_BAUDRATE | Baud rate coefficient (3) | (Wait 1mS) Operation status | Set communication baud rate |
| 03H | ENTER_SLEEP | Sleep mode (2) | Recognized value | Set the chip to go to sleep immediately |
| 05H | RESET_ALL | | (Wait 50mS） | Execute hardware reset |
| 06H | CHECK_EXIST | Any data | Invert by bit | Test the communication interface and operation status |
| 19H | GET_GLOB_INT_STATUS | | Global interrupt status (2) | Get global interrupt status Use CH392T |
| 21H | SET_MAC_ADDR | MAC address (6) | | Set MAC address |
| 22H | SET_IP_ADDR | IP address (4) | | Set IP address |
| 23H | SET_GWIP_ADDR | Gateway address (4) | | Set gateway IP address |
| 24H | SET_MASK_ADDR | Subnet mask (4) | | Set subnet mask |
| 26H | GET_PHY_STATUS | | PHY status | Get PHY status |

| | | | | |
|---|---|---|---|---|
| 27H | INIT_CH392 | | | CH392 initialization |
| 28H | GET_UNREACH_IPPORT | | Unreachable information (8) | Get unreachable IP, port, and protocol |
| 29H | GET_GLOB_INT_STATUS | | Global interrupt status (1) | Get global interrupt status Use CH392F |
| 2AH | SET_RETRAN_COUNT | Timeout retransmissions (2) | | Set TCP timeout retransmission count |
| 2BH | SET_RETRAN_PERIOD | Timeout retransmissions (2) | | Set TCP timeout retransmission period |
| 2CH | GET_CMD_STATUS | | Command execution status | Get command execution status |
| 2DH | GET_REMOT_IPP_SN | Socket index | IP and port (6) | Get remote (destination) IP and port |
| 2EH | CLEAR_RECV_BUF_SN | Socket index | | Clear the receive buffer of Socket |
| 2FH | GET_SOCKET_STATUS_SN | Socket index | Socket status | Get Socket status |
| 30H | GET_INT_STATUS_SN | Socket index | Socket interrupt | Gets the Socket interrupt status |
| 31H | SET_IP_ADDR_SN | Socket index<br>Destination IP (4) | | Set the destination IP address of Socket |
| 32H | SET_DES_PORT_SN | Socket index<br>Destination port (2) | | Set the destination port of Socket |
| 33H | SET_SOUR_PORT_SN | Socket index<br>Source port (2) | | Set the source port of Socket |
| 34H | SET_PROTO_TYPE_SN | Socket index<br>Protocol type | | Set the source port of Socket |
| 35H | OPEN_SOCKET_SN | Socket index | | Open Socket |
| 36H | TCP_LISTEN_SN | Socket index | | Enable Socket listening |
| 37H | TCP_CONNECT_SN | Socket index | | Enable Socket connection |
| 38H | TCP_DISCONNECT_SN | Socket index | | Disconnect Socket TCP |
| 39H | WRITE_SEND_BUF_SN | Socket index<br>Length (2)<br>Data stream (N) | | Send buffer write data to the Socket |
| 3BH | GET_RECV_LEN_SN | Socket index | Length (2) | Gets the length of data received by Socket |
| 3CH | READ_RECV_BUF_SN | Socket index<br>Length (2) | Data stream (N) | Receives data from the Socket receive buffer |
| 3DH | CLOSE_SOCKET_SN | Socket index | | Disable Socket |
| 40H | GET_MAC_ADDR | | MAC address (6) | Get MAC address |
| 41H | DHCP_ENABLE | Enable flag | | Enable (stop) DHCP |
| 42H | GET_DHCP_STATUS | | DHCP status | Get DHCP status |
| 43H | GET_IP_INF | | IP and other | Get IP, MASK, DNS and other |

| | | | information | information |
|---|---|---|---|---|
| 51H | SET_TTL | Socket index<br>TTL | | Set TTL value, up to 128 |
| 52H | SET_RECV_BUF | Socket index<br>Starting block address<br>Number of blocks | | Set the Socket receive buffer |
| 56H | SET_KEEP_LIVE_IDLE | 4-byte time parameter | | Set KEEPLIVE idle time |
| 57H | SET_KEEP_LIVE_INTVL | 4-byte time parameter | | Set the KEEPLIVE timeout |
| 58H | SET_KEEP_LIVE_CNT | Number of retries | | Set the timeout retry times of KEEPLIVE |
| 59H | SET_KEEP_LIVE_SN | Socket index<br>Configuration | | Set Socket KEEPLIVE |
| 60H | GET_SEND_QUEUE | | Number of available queues | Query the number of available send queues |
| 61H | SET_EXT_PARA | Set extended parameters (4) | | Set extended parameters |
| BAH | SET_MACRAW | | | Enter the MACRAW mode, send and receive MAC frames |

The command in the shadow part of the table usually needs to be executed for a certain period of time and the execution status of the command is queried. The MCU can obtain the status by running the GET_CMD_STATUS command. (Refer to CH392INC.H for status definition)

## 5.1. CMD_GET_IC_VER

This command is used to obtain the chip and firmware version. The 1 byte of data returned is the version number.

## 5.2. CMD_SET_BAUDRATE

This command is used to set the baud rate of the serial port communication on the CH392. If the CH392 works in serial communication mode, the default baud rate is 9600bps after the CH392 is reset. If the single-chip microcomputer supports a high communication speed, you can dynamically adjust the baud rate of serial communication by using this command. This command requires the input of three data, baud rate coefficient 0, Baud rate coefficient 1, and Baud rate coefficient 2. The following table is the corresponding relationship with the baud rate, the calculation formula:

BaudRate = (Baud rate coefficient 2 <<16) + (Baud rate coefficient 1 << 8) + Baud rate coefficient 0.

| Baud rate coefficient 2 | Baud rate coefficient 1 | Baud rate coefficient 0 | Serial communication Baud rate (bps) |
|---|---|---|---|
| 00H | 12H | C0H | 4800 |
| 00H | 25H | 80H | 9600 |
| 00H | 4BH | 00H | 19200 |
| 00H | 96H | 00H | 38400 |
| 00H | E1H | 00H | 57600 |
| 01H | 2CH | 00H | 76800 |

| | | | |
|---|---|---|---|
| 01H | C2H | 00H | 115200 |
| 07H | 08H | 00H | 460800 |
| 0FH | 42H | 40H | 1000000 |
| 1EH | 84H | 80H | 2000000 |

① The CH392F internal serial frequency (F_uart) is 32M and 36.923M. The chip switches the appropriate frequency according to the set baud rate. The formula for calculating the frequency division coefficient DL is DL=F_uart/8/bps, DL is rounded. Note the error when setting the baud rate.

② The internal serial frequency (F_uart) of the CH392T is fixed at 144M, and the frequency division coefficient DL is calculated as follows: DL=F_uart/16/bps, DL is rounded. Note the error when setting the baud rate.

Under normal circumstances, the serial port communication baud rate is set within 1mS, and the CH392 outputs the operation status with the newly set communication baud rate after completion, so the MCU should adjust its communication baud rate in time after issuing commands.

## 5.3. CMD_ENTER_SLEEP

This command puts CH392 into sleep mode. This command requires the input of 2 bytes of configuration value. Configuration word 0 sets the sleep mode, 01H is for light sleep and 02H is for deep sleep, and configuration word 1 is for specific sleep function configuration. The definitions are as follows:

| Configuration word 0 | Configuration word 1 | Function description |
|---|---|---|
| 01H | 00H | Light sleep mode 1:<br>Enter light sleep, turn off MAC and PHY<br>Wake-up sources: Serial commands (serial mode), SPI commands (SPI mode) |
| | 01H | Light sleep mode 2:<br>Enter light sleep, keep MAC and PHY<br>Wake-up sources: Serial commands (serial mode), SPI commands (SPI mode), PHY connection interrupt |
| | 03H | Light sleep mode 3:<br>Enter light sleep, keep MAC and PHY, can be woken up by magic frames<br>Wake-up source: Serial command (serial mode), SPI command (SPI mode), PHY connection interrupt, receive corresponding magic frame |
| 02H | 00H | Deep sleep mode:<br>Wake-up source: Arbitrary data received at serial port (serial port mode), CS pulled low (SPI mode) |

This command is only supported by 392T, version 0x18 and above.

The 392T will reply with an acknowledgement value of 0x57 when it receives the sleep command, after which the 392T needs at least 20uS to go to sleep. Do not send any command code to the 392T within 20uS after receiving the sleep command to prevent accidental wakeup.

Upon awakening from light sleep mode, the 392T does not have to experience wait times. While sleeping, the 392T is able to parse and wake up normally to any command received.

After waking up from deep sleep mode, the 392T requires 2 to 3 mS to complete the wake-up process. During this wake-up phase, the 392T does not parse or respond to any received command codes. Therefore, do not send any

command codes to the 392T while it is being woken up. It is recommended that the user wake up 4mS before operating the 392T.

## 5.4. CMD_RESET_ALL

This command enables the CH392 to reset hardware. Normally, the hardware reset is completed in less than 50mS.

## 5.5. CMD_CHECK_EXIST

This command is used to test the communication interface and operating status to check whether the CH392 works properly. This command requires one byte of data, which can be any data. If the CH392 works properly, the output data of the CH392 is the inverse of the input data. For example, if the input data is 57H, the output data is A8H.

## 5.6. CMD_GET_GLOB_INT_STATUS_ALL

This command is used or the global interrupt status of the chip, only used by CH392T. After receiving this command, the chip will output 2 bytes of data, the low byte is in the front and the high byte is in the back, the global status is defined as follows:

| Bit | Name | Description |
|---|---|---|
| 15:12 | Reserved | |
| 11 | GINT_STAT_SOCK7 | Socket7 interrupt |
| 10 | GINT_STAT_SOCK6 | Socket6 interrupt |
| 9 | GINT_STAT_SOCK5 | Socket5 interrupt |
| 8 | GINT_STAT_SOCK4 | Socket4 interrupt |
| 7 | GINT_STAT_SOCK3 | Socket3 interrupt |
| 6 | GINT_STAT_SOCK2 | Socket2 interrupt |
| 5 | GINT_STAT_SOCK1 | Socket1 interrupt |
| 4 | GINT_STAT_SOCK0 | Socket0 interrupt |
| 3 | GINT_STAT_DHCP | DHCP interrupt |
| 2 | GINT_STAT_PHY_CHANGE | PHY Status change interrupt |
| 1 | GINT_STAT_IP_CONFLI | IP conflict |
| 0 | GINT_STAT_UNREACH | Unreachable interrupt |

① GINT_STAT_UNREACH: indicates unreachable interrupt. After receiving the ICMP unreachable interrupt packet, the CH392 saves the IP address, port, and protocol type of the unreachable IP packet in the unreachable information table. After receiving the interrupt, the microcontroller sends the GET_UNREACH_IPPORT command to obtain the unreachable information.

② GINT_STAT_IP_CONFLI: IP conflict interrupted. This interrupt is generated when the CH392 detects that its IP address is the same as that of other network devices in the same network segment.

③ GINT_STAT_PHY_CHANGE: indicates that the PHY change is interrupted. This interrupt occurs when the PHY connection of the CH392 changes. For example, the PHY status changes from connected to disconnected or from disconnected to connected. The microcontroller can send the GET_PHY_STATUS command to get the status of the current PHY connection.

④ GINT_STAT_DHCP: DHCP interrupt If the DHCP function of CH392 is enabled on the microcontroller, the CH392 will generate this interrupt. The microcontroller can send the CMD_GET_DHCP_STATUS command to obtain the DHCP status. If the status is 0, it indicates that the DHCP status is successful.

⑤ GINT_STAT_SOCK0 - GINT_STAT_SOCK7: indicates that the Socket is interrupted. When the Socket has an interrupt event, CH392 generates this interrupt event. The MCU needs to send GET_INT_STATUS_SN to obtain the Socket interrupt status. See the GET_INT_STATUS_SN section.

After this command is executed, CH392 sets the INT# pin to high and clears the global interrupt status.

### 5.7. CMD_SET_MAC_ADDR

This command is used to set the MAC address of the CH392. You need to enter a six-byte MAC address, with the lowest byte starting. The parameters must be set before the initialization command.

The MAC address assigned by IEEE has been programmed before the CH392 chip is delivered. Do not set a MAC address unless necessary.

### 5.8. CMD_SET_IP_ADDR

This command is used to set the IP address of the CH392. You need to enter a four-byte IP address, with the lowest byte starting. In this document, the byte order of all commands that contain IP input or output is the lowest byte of IP.

### 5.9. CMD_SET_GWIP_ADDR

This command is used to set the gateway address of the CH392. You need to enter a 4-byte IP address.

### 5.10. CMD_SET_MASK_ADDR

This command is used to set the subnet mask of the CH392. You need to enter a mask of 4-byte. The default value is 255.255.255.0. This parameter is optional.

### 5.11. CMD_GET_PHY_STATUS

This command is used to query the connection status of the PHY. After receiving this command, CH392 queries the current PHY connection status and outputs a 1-byte PHY connection status code:

If the connection status code is 01H, the PHY connection is disconnected.

If the connection status code is 02H, the PHY connection is 10M full-duplex.

If the connection status code is 04H, the PHY connection is 10M half-duplex.

### 5.12. CMD_INIT_CH392

This command is used to initialize the MAC address, PHY, and TCP/IP protocol stack of the CH392. This command takes about 350mS to complete. The MCU can send GET_CMD_STATUS to check whether the execution is complete and the execution status.

### 5.13. CMD_GET_GLOB_INT_STATUS

This command is used to query the global interrupt status, only used by CH392T. After receiving this command, the CH392 outputs the global interrupt status with 1 byte. The global interrupt status is defined as follows:

| Bit | Name | Description |
|---|---|---|
| 7 | GINT_STAT_SOCK3 | Socket3 interrupt |
| 6 | GINT_STAT_SOCK2 | Socket2 interrupt |
| 5 | GINT_STAT_SOCK1 | Socket1 interrupt |
| 4 | GINT_STAT_SOCK0 | Socket0 interrupt |
| 3 | GINT_STAT_DHCP | DHCP interrupt |

| | | |
|---|---|---|
| 2 | GINT_STAT_PHY_CHANGE | PHY status change interrupt |
| 1 | GINT_STAT_IP_CONFLI | IP conflict |
| 0 | GINT_STAT_UNREACH | Unreachable interrupt |

① GINT_STAT_UNREACH: indicates unreachable interrupt. After receiving the ICMP unreachable interrupt packet, the CH392 saves the IP address, port, and protocol type of the unreachable IP packet in the unreachable information table. After receiving the interrupt, the microcontroller sends the GET_UNREACH_IPPORT command to obtain the unreachable information.

② GINT_STAT_IP_CONFLI: IP conflict interrupted. This interrupt is generated when the CH392 detects that its IP address is the same as that of other network devices in the same network segment.

③ GINT_STAT_PHY_CHANGE: indicates that the PHY change is interrupted. This interrupt occurs when the PHY connection of the CH392 changes. For example, the PHY status changes from connected to disconnected or from disconnected to connected. The MCU can send the GET_PHY_STATUS command to obtain the status of the current PHY connection.

④ GINT_STAT_DHCP: DHCP interrupt If the DHCP function of CH392 is enabled on the microcontroller, the CH392 will generate this interrupt. The microcontroller can send the CMD_GET_DHCP_STATUS command to obtain the DHCP status. If the status is 0, it indicates that the DHCP status is successful.

⑤ GINT_STAT_SOCK0 - GINT_STAT_SOCK3: indicates that the Socket is interrupted. When the Socket has an interrupt event, CH392 generates this interrupt event. The MCU needs to send GET_INT_STATUS_SN to obtain the Socket interrupt status. See the GET_INT_STATUS_SN section.

After this command is executed, CH392 sets the INT# pin to high and clears the global interrupt status.

## 5.14. CMD_SET_RETRAN_COUNT

This command is used to set the TCP timeout retransmission number, you need to enter the number of retries in 2 bytes (the low byte comes first), the maximum value is 31. TCP timeout retransmission number refers to the number of times the 392T will try to retransmit a packet after the CH392 sends out a packet outward and does not receive an acknowledgement packet back from the device at the opposite end, the number of times it will attempt to retransmit this packet. After attempting the set number of times and still not receiving an acknowledgement packet from the opposite end, the transmission is considered to have failed and a TIME_OUT interrupt is reported.

This command must be configured before the CH392 is initialized, and the default value is 20, which generally does not need to be set by the user.

## 5.15. CMD_SET_RESET_RETRAN_PERIOD

This command is used to set the tcp timeout retransmission period, you need to enter a 2-byte retry period value (the low byte comes first), the maximum value is 31, the unit is 50ms. the smaller the value, the higher the detection frequency, the faster the retransmission is relative; the larger the value, the lower the detection frequency, the slower the retransmission is relative. However, the specific retransmission time is decided by the CH392T based on the network congestion and other factors.

This command must be configured before the CH392 is initialized, and the default value is 10 (500ms), which generally does not need to be set by the user.

## 5.16. CMD_GET_CMD_STATUS

This command is used to query the execution status of a command. The CH392 outputs 1 byte of data indicating the status of the command execution. The command execution status is as follows:

| Code | Name | Description |
|---|---|---|
| 00H | CH392_ERR_SUCCESS | Success |
| 10H | CH392_ERR_BUSY | Busy: The command is being executed |
| 11H | CH392_ERR_MEM | Memory management error |
| 12H | CH392_ERR_BUF | Buffer error |
| 13H | CH392_ERR_TIMEOUT | Timeout |
| 14H | CH392_ERR_RTE | Routing error |
| 15H | CH392_ERR_ABRT | Connection abort |
| 16H | CH392_ERR_RST | Connection reset |
| 17H | CH392_ERR_CLSD | Connection close |
| 18H | CH392_ERR_CONN | connectionless |
| 19H | CH392_ERR_VAL | Value error |
| 1AH | CH392_ERR_ARG | Parameter error |
| 1BH | CH392_ERR_USE | Have been used |
| 1CH | CH392_ERR_IF | MAC error |
| 1DH | CH392_ERR_ISCONN | Connected |
| 20H | CH392_ERR_OPEN | Open |

If the microcontroller receives CH392_ERR_BUSY, it indicates that the CH392 is executing commands. The microcontroller should send the CMD_GET_CMD_STATUS command at least 2 milliseconds later to obtain the status.

All commands in the shaded portion of the command code table require CMD_GET_CMD_STATUS to be sent to obtain the execution status.

## 5.17. CMD_GET_REMOT_IPP_SN

This command is used to obtain the IP address and port number of the remote end. You need to enter the Socket index value of 1 byte. The CH392 output 4 bytes of the IP address and 2 bytes of the port number. After the Socket works in TCP Server mode and the connection is established, the MCU can use this command to obtain the IP address and port number of the remote end.

## 5.18. CMD_CLEAR_RECV_BUF_SN

This command is used to empty the receive buffer of the Socket. You need to enter a one-byte Socket index value. Upon receiving this command, the CH392 clears the receive length of the Socket to zero. The cleared data generally includes only the cache that the interface has reported to be processed, but does not include the data that has not been reported in the Socket protocol processing. If there is unreported data in the protocol processing after the clear operation is performed, the corresponding socket will once again generate an acceptance buffer out-of-air break.

## 5.19. CMD_GET_SOCKET_STATUS_SN

This command is used to obtain the Socket status. You need to enter a one-byte Socket index value. After receiving this command, the CH392 outputs a two-byte status code.

The first status code is the status code of the Socket, which is defined as follows:

| Code | Name |
|---|---|
| 00H | SOCKET_CLOSED |

| | |
|---|---|
| 05H | SOCKET_OPEN |

The second status code is the status code of TCP, which is meaningful only when the TCP mode has been opened. The TCP status code is defined as follows:

| Code | Name | Description |
|---|---|---|
| 00H | TCP_CLOSED | Closed |
| 01H | TCP_LISTEN | Listen |
| 02H | TCP_SYN_SENT | SYN sent |
| 03H | TCP_SYN_RCVD | SYN received |
| 04H | TCP_ESTABLISHED | TCP connection established |
| 05H | TCP_FIN_WAIT_1 | The active closing side first sends FIN |
| 06H | TCP_FIN_WAIT_2 | The active closing side receives an ACK from FIN |
| 07H | TCP_CLOSE_WAIT | The passive closing side receives FIN |
| 08H | TCP_CLOSING | Closing |
| 09H | TCP_LAST_ACK | The passive closing side sends FIN |
| 0AH | TCP_TIME_WAIT | 2MLS wait status |

TCP status are defined in TCP/IP. For details, see TCP/IP.

Only the following items are required: TCP_CLOSED, TCP_LISTEN, and TCP_ESTABLISHED. Other status chips are processed and updated automatically.

## 5.20. CMD_GET_INT_STATUS_SN

This command is used to obtain the Socket interrupt status. You need to enter a one-byte Socket index value. After receiving this command, CH392 outputs a one-byte Socket interrupt code.

| Bit | Name | Description |
|---|---|---|
| 7 | - | Reserved |
| 6 | SINT_STAT_TIM_OUT | Timeout |
| 5 | - | Reserved |
| 4 | SINT_STAT_DISCONNECT | TCP disconnect |
| 3 | SINT_STAT_CONNECT | TCP connect |
| 2 | SINT_STAT_RECV | Receive buffer not empty |
| 1 | - | Reserved |
| 0 | SINT_STAT_SENBUF_FREE | Send buffer free |

① SINT_STAT_SENBUF_FREE: Send buffer idle interrupt. After the single chip computer sends data to the Socket buffer, CH392 quickly copies the data to the internal protocol stack or MAC buffer to encapsulate the data. When the data replication is complete, this interrupt is generated. The MCU can continue to write subsequent data to the send buffer. After the single chip microcomputer sends data to the Socket buffer once, it must wait until this interrupt is generated before it can write the next data. CH392 uses queue management to send data. When the number of sockets in use exceeds the number of queues, the CH392 sends data after waiting for "Send buffer Idle Interrupt" to check whether the number of queues is available, and then starts to send data. The master can simplify the operation and directly determine the number of queues is available.

② SINT_STAT_CONNECT: indicates that the TCP connection is interrupted. This parameter is valid only in TCP mode. It indicates that the TCP connection is successful, and the single chip computer can transmit data only after the interruption is generated.

③ SINT_STAT_DISCONNECT, indicates that the TCP connection is disconnected. This parameter is valid only

in TCP mode, indicating that the TCP connection is disconnected.

④ SINT_STAT_TIM_OUT: indicates that the timeout occurs during TCP connection, disconnection, or data sending in TCP mode.

After TCP timeout "SINT_STAT_TIM_OUT" and disconnection "SINT_STAT_DISCONNECT", the Socket will be actively closed, and the cache data will still be retained, so that the MCU can read the remaining data in time after TCP disconnection. However, if TCP connection occurs again, the cache will be completely cleared.

## 5.21. CMD_SET_IP_ADDR_SN

This command is used to set the destination IP address of the Socket. You need to enter a one-byte Socket index value and a four-byte destination IP address. If the Socket works in UDP or TCP Client mode, you must set the destination IP address before sending the CMD_OPEN_SOCKET_SN command.

## 5.22. CMD_SET_DES_PORT_SN

This command is used to set the destination port of the Socket. You need to enter the Socket index value of 1 byte and the destination port of 2 bytes (the lowest byte is the first). If the Socket works in UDP or TCPClient mode, this parameter must be set.

## 5.23. CMD_SET_SOUR_PORT_SN

This command is used to set the Socket source port. You need to enter the Socket index value of 1 byte and the source port of 2 bytes (the lowest byte is the first). If two or more sockets use the same mode, the source port number must not be the same. For example, if Socket0 is in UDP mode and the source port is 600, and Socket1 is also in UDP mode, do not use source port 600. Otherwise, the port may fail to open.

## 5.24. CMD_SET_PROTO_TYPE_SN

This command is used to set the Socket working mode. You need to enter a one-byte Socket index value and one-byte working mode. The working mode is defined as follows:

| Code | Name | Description |
|---|---|---|
| 03H | PROTO_TYPE_TCP | TCP mode |
| 02H | PROTO_TYPE_UDP | UDP mode |
| 0AH | PROTO_TYPE_PING | PING mode |

This command must be run before CMD_OPEN_SOCKET_SN. For details, see 8.2 Application Reference.

## 5.25. CMD_OPEN_SOCKET_SN

This command is used to open a Socket. To use a Socket, you need to enter a one-byte Socket index. After sending this command, the MCU should send GET_CMD_STATUS to check the command execution status. After opening the Socket and returning success, the data transfer can proceed. Before running this command, you must set the destination IP address, protocol type, source port, and destination port. For details, see 8.2 Application Reference.

## 5.26. CMD_TCP_LISTEN_SN

This command is valid only in TCP mode, enabling the Socket to enter the listening mode, that is, the TCPServer mode. You need to enter a one-byte Socket index. This command must be executed after OPEN_SOCKET_SN. After sending this command, the MCU should send GET_CMD_STATUS to check the execution status of the command.

In TCP Server mode, the Socket detects connection events until the connection is successful, resulting in a SINT_STAT_CONNECT interrupt. Only one connection can be established per Socket. If a qualifying connection event is received again, the Socket sends TCPRESET to the remote end trying to connect.

## 5.27. CMD_TCP_CONNECT_SN

This command is valid only in TCP mode. To enable the Socket to enter the connection mode (TCP Client mode), you need to enter a one-byte Socket index value. After sending this command, the MCU should send GET_CMD_STATUS to check the execution status of the command.

Upon receipt of this command, the Socket will initiate a connection event. If the connection succeeds, a SINT_STAT_CONNECT interrupt will occur. If an exception occurs during the connection or the connection fails after a specified period of time, the SINT_STAT_TIM_OUT function is interrupted. The MCU receives this interrupt, and if it needs to connect again, it needs to reopen the Socket and perform TCP_CONNECT_SN.

## 5.28. CMD_TCP_DISCONNECT_SN

This command is valid only in TCP mode. To disconnect the current TCP connection, you need to enter a one-byte Socket index value. After sending this command, the MCU should send GET_CMD_STATUS to query the execution status of the command. After the current TCP connection is successfully disconnected, SINT_STAT_DISCONNECT interrupt is displayed.

## 5.29. CMD_WRITE_SEND_BUF_SN

This command is used to write data to the send buffer of the Socket. You need to enter the Socket index value of 1 byte, the length of 2 bytes (the low byte is the first), and several bytes of data stream. The length of the input data must not be greater than the size of the send buffer, when the external microcontroller writes the data, CH392 will be based on the working mode of the Socket packet encapsulation, and then send it.

Send cache description:

The CH392 has several send cache queues for sending Socket data. Run the CMD_WRITE_SEND_BUF_SN command to store the sent index number, Send Length, and Send data in an idle queue. After finding available queues, the CH392 starts encapsulation and sending. The CH392T has four queues and the CH392F has two queues. The queue cache has 1536 bytes, but the amount of data sent should not exceed the effective data load of the protocol maximum frame. After performing a send, wait until the current socket send buffer is free before performing the next send.

Different modes for writing cache data processing:

TCP mode: After receiving the write data command, the data in the queue is subcontracted according to TCPMSS or the current internal TCP available cache, and the cache empty flag is set after all data is sent.

UDP mode: The maximum UDP mode only processes 1472 bytes, that is, the maximum frame of UDP mode. It is recommended that the master do subcontracting when sending data, and set the cache empty flag after the completion of single packet sending.

## 5.30. CMD_GET_RECV_LEN_SN

This command is used to get the valid data length of the current receive buffer, which requires the input of a 1-byte socket index value, and the CH392 outputs a 2-byte length (with the low byte in front) when it receives this command.

## 5.31. CMD_READ_RECV_BUF_SN

This command is used to read data from the Socket receive buffer. You need to enter the Socket index value of 1 byte and the length of 2 bytes (the lower byte is the first). CH392 outputs a data stream of several bytes according to the length value. In practice, you can first send the RECV_LEN_SN command to obtain the actual effective length of the current buffer. The length of the read data can be less than the actual valid data length of the buffer (for detailed steps, please refer to 8.2 Application Reference Steps), and the unread data is still retained in the receive buffer, which can be continued to be read by the microcontroller through this command.

## 5.32. CMD_CLOSE_SOCKET_SN

This command is used to close a socket, and requires a 1-byte socket index value. After closing the socket, the data in the socket's receive-send buffer will still be retained, and the socket configuration information will still be retained after the socket is closed, so you only need to open the socket again the next time you use the socket.

In TCP mode, the CH392 will automatically disconnect the TCP connection before closing the socket.

CH392T version 0x18 and before do not need to query the command execution status after closing the socket, you can query the socket status by the GET_SOCKET_STATUS_SN command to confirm whether it is successfully closed or not.

## 5.33. CMD_GET_MAC_ADDR

This command is used to obtain a MAC address. After receiving this command, the CH392 outputs a MAC address of 6 bytes.

## 5.34. CMD_DHCP_ENABLE

This command is used to enable or stop DHCP. You need to enter a one-byte flag code. If the flag code is 1, the DHCP is enabled. If the flag code is 0, the DHCP is disabled.

After DHCP is enabled, the CH392 broadcasts a DHCP DISCOVER message to the network to discover the DHCP Server, request an address and other configuration parameters after finding the DHCP Server, and generate GINT_STAT_DHCP interrupt. After this interrupt, the mono can send the command GET_DHCP_STATUS to obtain the DHCP status. If the status is 0, it indicates success. The MCU can send the GET_IP_INF command to obtain information such as IP address and MASK. If the status is 1, it indicates an error, which is usually caused by timeout. For example, the DHCP Server is not found.

After DHCP is started, it is always in the operating status, unless the command to disable DHCP is received from the MCU. During this process, if the DHCP Server assigns a new configuration to the CH392 and the configuration is different from the original, the CH392 still generates an interrupt.

If the DHCP Server is not discovered, the CH392 continues to send a DHCP DISCOVER message at an interval of 16 seconds.

This command takes about 20mS to execute. The MCU can send GET_CMD_STATUS to check whether the execution is complete and the execution status is complete.

## 5.35. CMD_GET_DHCP_STATUS

This command is used to obtain the DHCP status. Generally, after the microcontroller receives an interrupt, this command is used to obtain the DHCP execution status. After receiving this command, the CH392 outputs the DHCP status codes, which are 0 and 1 respectively.

If the status is 0, it indicates success, then the MCU can send the command GET_IP_INF to obtain IP, MASK and other information.
If the status is 1, it indicates an error, which is usually caused by a timeout. For example, the DHCPServer is not found.

## 5.36. CMD_GET_IP_INF

This command is used to obtain information such as IP, GatewayIP, MASK, and DNS. After receiving this command, the CH392 outputs 20 bytes of data in sequence, as follows: 4-byte IP address, 4-byte gateway IP address, 4-byte subnet mask, 4-byte DNS1(primary DNS), 4-byte DNS2(secondary DNS).
After DHCP, you can send this command to obtain information about the current CH392. If some configurations are not obtained in DHCP, the configurations are set to 0. For example, DNS is not always allocated when DHCP is used on the LAN. When this command is used to obtain configuration information, both DNS1 and DNS2 are set to 0.

## 5.37. CMD_SET_TTL

This command is used to set the TTL of a Socket. You need to enter a 1-byte Socket index and a 1-byte TTL value. By default, it is not required. If it is set, it should be set after opening the Socket. The maximum value is 128.

## 5.38. CMD_SET_RECV_BUF

1. The CH392T receive cache is configured as follows:
This command is used to set the receive buffer of a Socket. You need to enter 3 bytes of data. The first byte is the Socket index, the second byte is the start block of the buffer, and the third byte is the number of blocks.

| | | | | | |
|---|---|---|---|---|---|
| Block 0 | Block 1 | Block 2 | …… | Block 63 | Block 64 |

The CH392T internal receive buffer structure is shown in the figure above. It consists of 64 blocks, each of which is 512 bytes long. The single chip microcomputer can freely allocate the size of each Socket receive buffer. When setting the receive buffer, note that the blocks are contiguous, and the caches allocated by different sockets should be independent and cannot be overlapped. After initialization, the CH392T allocates the buffer as follows:

| Socket | Starting block | Block number | Buffer size |
|---|---|---|---|
| 0 | 0 | 8 | 4KB |
| 1 | 8 | 8 | 4KB |
| 2 | 16 | 8 | 4KB |
| 3 | 24 | 8 | 4KB |
| 4 | 32 | 8 | 4KB |
| 5 | 40 | 8 | 4KB |
| 6 | 48 | 8 | 4KB |
| 7 | 56 | 8 | 4KB |

As can be seen from the above table, CH392T allocates all buffers to Socket0-7 after initialization, and each Socket receives 8 blocks (4KB) of buffer.

2. Setting the CH392F Receive cache:
The command structure is the same as above. CH392F receives 16 blocks of cache, each of which is 536 bytes. After initialization, CH392F allocates four blocks to each Socket of Socket0-3. If the application has requirements for a specific Socket receive, the number of receive buffer blocks can be reallocated.

3.UDP SERVER Mode Cache Description (CH392T):

When the Socket is set to UDP Server mode (that is, the target IP address is 255.255.255.255 and UDP frames are received on the port), the CH392T Socket cache is divided into two parts. The first part is used for low-level Ethernet receiving processing, and the second part is used for cache and interface communication. Partition method: The cache of the current allocated Socket is greater than or equal to 3 times the maximum UDP load length (1472 bytes), then a fixed UDP maximum load length cache is allocated to Ethernet receiving processing, and the rest is all used for interface communication cache; If the cache of the currently allocated Socket is less than 3 times the maximum UDP load length, the entire cache is divided into 3 equal parts, 1/3 cached for Ethernet receiving processing, and the remaining 2/3 cached for interface communication. When using UDPSERVER mode, it is recommended that the cache be allocated at least 4K and that the maximum UDP load length be more than 3 times. Note that the maximum data frame length to be received does not exceed the Ethernet receive cache length.

## 5.39. CMD_SET_KEEP_LIVE_IDLE

This command is used to set the idle time of KeepLive. You need to enter a time value of 4 bytes, in milliseconds. This command applies only to TCP connections. KeepLive idle time refers to the time between sending KeepLive packets and receiving no data on the TCP connection. The default value is 20000. In practical applications, there is a certain relationship between the KeepLive idle time and the TCP timeout retransmission period, and usually the actual idle time is about 1 to 1.2 times of the KeepLive idle time setting.

## 5.40. CMD_SET_KEEP_LIVE_INTVL

This command is used to set the KeepLive timeout time. you need to enter a 4-byte time value in milliseconds. This command is only used for TCP connections. the KeepLive timeout time is the time to wait for an answer after a KeepLive packet is sent. The default value is 15000.

## 5.41. CMD_SET_KEEP_LIVE_CNT

This command is used to set the timeout times of KeepLive. You need to enter the timeout times of 1 byte. This command applies only to TCP connections. The KeepLive timeout count is the maximum number of consecutive times that KeepLive packets are allowed to go unanswered. The default value is 9.

Assume that the IDLE time of KeepLive is IDLE, the timeout time is INTVL, and the timeout times are CNT.

If the KeepLive function is enabled on the Socket connection, the Socket will start sending KeepLive packets when the TCP connection is IDLE (no data is sent or received) for mS. If the remote end answers ACK within INTVL mS, the connection is considered normal. Otherwise, the Socket considers that the socket times out after INTVL milliseconds and starts sending KeepLive packets again. If no ACK packet is received within CNT times, the current connection is considered disconnected. The SINT_STAT_TIM_OUT or SINT_STAT_DISCONNECT interrupt is generated.

If IDLE, INTVL, and CNT need to be configured, IDLE must be greater than INTVL and the value is a multiple of 500.

## 5.42. CMD_SET_KEEP_LIVE_SN

This command is used to enable or disable the KeepLive function of a Socket. You need to enter a 1-byte Socket index and a 1-byte configuration value. The configuration value 0 indicates that the KeepLive function of a Socket is disabled, and 1 indicates that the Keeplive function of a socket is enabled. The default value is off.

When the Socket is a TCP client, run this command to enable the KeepLive function after creating the Socket.

When the Socket is a TCP server, use this command to enable the KeepLive function after

SINT_STAT_CONNECT is generated.

## 5.43. CMD_SET_MACRAW

This command is used to enable the CH392T (CH392F invalid) to be used in MACRAW mode to send and receive data at the MAC layer. In this case, the Socket parameter in the command is meaningless and does not have the meaning of the actual Socket existence. Therefore, you cannot close or open the Socket. Socket0 must be used as the operation number.

If the CH392T is in MACRAW mode, you do not need to initialize the CH392T. After the CH392T is reset, the CH392T automatically initializes and enters MACRAW mode.

## 5.44. CMD_GET_UNREACH_IPPORT

This command is used to obtain unreachable information, including IP address, port number, and protocol type. After receiving an unreachable packet, the CH392 generates an unreachable interrupt. The single chip microcomputer can use this command to obtain unreachable information. After receiving this command, the CH392 will output 1 byte unreachable code, 1 byte protocol type, 2 bytes port number (the low byte is the first), and 4 bytes IP. The single chip computer can judge whether the protocol is unreachable, the port is unreachable or the IP is unreachable according to the unreachable code. Please refer to the RFC792 standard for unreachable code or refer to the sample program.

## 5.45. CMD_SET_EXT_PARA

This command is only supported by 392T, version 0x1A and above.

This command is used to configure the 392T extended parameters, you need to enter 4-byte parameters, the meaning of each parameter is as follows:

| Bit | Name | Description |
|---|---|---|
| 31:3 | - | Reserved, must be 0 |
| 4:3 | TCP Socket shutdown mode configuration | This command can be configured at any moment and takes effect for all TCP sockets after configuration, the default is 02H.<br>00H: Configuration does not take effect;<br>01H: Normal shutdown, disconnect TCP connection after 4 waves, close Socket;<br>02H: RST shutdown, 392T sends RST message directly to the opposite end, disconnects the TCP connection, and closes the Socket;<br>03H:Direct shutdown, does not send any packet, and closes the Socket directly. |
| 2:1 | - | Reserved, must be 0 |
| 0 | - | Reserved, must be 1 |

This configuration value generally does not need to be set by the user.

## 5.46. CMD_GET_SEND_QUEUE

This command is used to obtain the number of available socket queues. The return value is the number of available socket queues. When the value is not 0, the single chip microcomputer can perform a sending process. It is recommended that the microcontroller query this parameter before sending each time and then execute the

sending process.

# 6. Functional Specification

## 6.1 MCU Communication Interfaces

There are two communication interfaces between CH392 and single chip microcomputer: SPI interface and UART. When the chip is powered on and reset, the CH392 will sample the SEL pin status and select the communication interface according to this pin level, refer to the following table (0 represents low level, 1 represents high level or suspended).

| SEL pin | Select communication interface |
|---|---|
| 1 | UART |
| 0 | SPI interface |

The interrupt request output of the CH392 chip INT# pin is low by default, and can be connected to the interrupt input pin of the MCU or the ordinary input pin. The MCU can use the interrupt mode or query mode to know the interrupt request of the CH392.

## 6.2 SPI Interfaces

SPI synchronous serial interface signal lines include: SPI slice select input pin SCS, serial clock input pin SCK, serial data input pin SDI, serial data output pin SDO. Through the SPI interface, CH392 can be connected to the SPI serial bus of various single-chip computers, DSPS and MCUS with fewer connections, or point-to-point connection at a longer distance.

The SPI clock frequency is 24M for the CH392T and 10M for the CH392F.

Transmission byte delay: CH392T about 1uS, CH392F about 3uS.

The SCS pin of the CH392 chip is driven by the SPI chip selective output pin or ordinary output pin of the single chip microcomputer, the SCK pin is driven by the SPI clock output pin of the single chip microcomputer SCK, and the SDI pin is driven by the SPI data output pin SDO or MOSI of the single chip microcomputer. The SDO pins are connected to the SPI data input pins SDI or MISO of the microcontroller. For hardware SPI interfaces, you are advised to set the SPI to CPOL=CPHA=0 or CPOL=CPHA=1, and set the sequence of data bits to MSB first. The SPI interface of CH392 also supports the communication of single chip microcomputer with ordinary I/O pin analog SPI interface.

The SPI interface of the CH392 supports SPI mode 0 and SPI mode 3. The CH392 always inputs data from the rising edge of the SPI clock SCK, and outputs data from the falling edge of the SCK when it is allowed to output. The data bit order is the highest in the front, and counting up to 8 bits is a byte.

The steps of SPI are:

① The single chip microcomputer generates SPI slice selection of CH392 chip, which is active low;

② The single chip microcomputer sends out a byte of data according to the SPI output mode, and CH392 always regards the first byte received after the SPI slice selection SCS is effective as the command code, and the following bytes as the data.

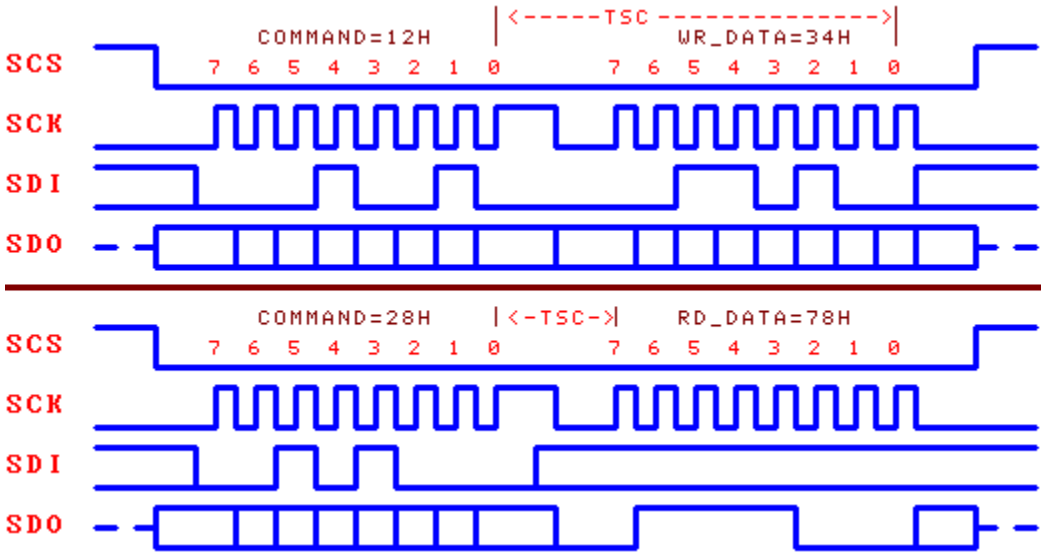
③ SCM delay TSC time (CH392T about 1uS, CH392F about 3uS) waiting for the SPI interface of CH392 to be idle;

④ If it is a write operation, the microcontroller sends a byte of data to CH392. After the SPI interface is free, the microcontroller continues to send several bytes of data to be written, and CH392 receives it successively until the microcontroller disables SPI slice selection.

⑤ If it is a read operation, the microcontroller receives one byte of data from CH392, waits for the SPI interface to be idle, and continues to receive several bytes of data from CH392 until the microcontroller disables SPI slice selection.

⑥ The single chip microcomputer disables the SPI chip selection of CH392 chip to end the current SPI operation.

The following is the logical sequence diagram of the SPI interface, the former is to issue the command 12H and write the data 34H, the latter is to issue the command 28H and read the data 78H.




## 6.3 UART

UART signal lines include: serial data input pin RXD and serial data output pin TXD. Through the serial interface, the CH392 can be connected to the MCU, DSP and MCU with a minimum of connections.

The RXD and TXD of CH392 chip can be connected to the serial data output pin and serial data input pin respectively.

The serial data format of the CH392 is a standard byte transfer mode, consisting of 1 start bit, 8 data bits, and 1 stop bit.

CH392 not only supports the hardware to set the default serial communication baud rate, but also supports the MCU to choose the appropriate communication baud rate at any time through the CMD_SET_BAUDRATE command. After each power-on and reset, the default serial communication baud rate of the CH392 is 9600bps.

In order to distinguish between command code and data, the CH392 requires the MCU to first send two synchronization code bytes (57H and ABH) through the serial port, and then send the command code, followed by sending data or receiving data. The CH392 checks the interval between the two synchronization code bytes and between the synchronization code and the command code. If the interval is longer than SER_CMD_TIMEOUT (about 40mS), the CH392 discards the synchronization code and command package. Perform the following steps to run serial port commands:

① The single chip microcomputer sends the first synchronization code 57H to CH392 through the serial port;
② SCM sends the second synchronization code ABH to CH392;
③ SCM issues command code to CH392;
④ If the command has input data, the input data is issued to CH392 one byte at a time.
⑤ If the command has output data, the output data is received from CH392 one byte at a time.
⑥ After the command is completed, some commands will generate an interrupt notification and the interrupt status code can be read through the serial port. The MCU can pause or go to ① to continue to execute the next command.

## 6.4 Other Hardware

The CH392 chip integrates 10M Ethernet PHY and MAC, SPI-Slave controller, UART, SRAM, high-speed MCU, firmware program, crystal oscillator and PLL frequency multiplier, power on reset circuit, etc.
The RXP, RXN, TXP and TXN of the CH392 chip are Ethernet signal lines. The PHY of the CH392 supports automatic MDI/MDIX line conversion.
The CH392 chip has a built-in power power-on reset circuit, and in general, no external reset is required. The RSTI pin is used to input the asynchronous reset signal from the outside; When the RSTI pin is low voltage, the CH392 chip is reset; When the RSTI pin returns to a high level, the CH392 will continue to delay the reset for about 35mS, and then enter the normal operating status. In order to reliably reset during power on and reduce external interference, a capacitor with a capacity of about 0.1uF can be connected between the RSTI pin and the ground.

## 6.5 PING Mode (Supported by CH392F)

CH392F Setting Socket mode can be set to PING. By using the socket to send and receive cache, you can control the CH392F to send ICMP PING frames. The cache operation mode is consistent with the normal mode.
Control command frame format definition:
< Sign > + < protocol version > + < PING interval > + < timeout > + < data type > + < data length >, < transmission times > + < command code > + < IP >
Reply command frame format definition:
< Sign > + < protocol version > + < commands and status > + < PING status > + <reception times > + < transmission times > + < reception time >

### 6.5.1 Parameter Definition

Control command frame parameters:

| Name | Description | format |
|---|---|---|
| Flag | Identifies the PING command frame.<br>Value: 0x57 0xAB | 2-byte flag |
| Protocol version | Format definitions used to distinguish between different versions, current version: 0x0100 | 2-byte of version information |
| PING interval | PING interval for sending frames. The value must be greater than the timeout period (mS) | 2-byte, small endian format |
| Timeout | PING Monitors the timeout parameter of the received reply frame after the transmission, in mS | 2-byte, small endian format |
| Data type | 1: Fixed 0xFF filling;<br>2: Fixed 0x00 filling;<br>3: Random data filling;<br>4: Cumulative data filling | 2-byte, small endian format |
| Transmission times | The transmitting times of the PING task are set. When the transmitting times of the ping task are set, the ping task ends automatically | 2-byte, small endian format |
| Command code | 0x0101 enable statistics mode, 0x0201 enable Echo Mode | 2-byte, small endian format |
| IP | PING the IP address of the peer end | 4-byte |

Reply command frame parameters:

| Name | Description | format |
|---|---|---|
| Command and status | Command code response or command code execution status | 2-byte, small endian format |
| PING status | 0x0000: Idle<br>0x0001: Normal operation<br>0x0002: Send exception<br>0x0003: PING timeout | 2-byte, small endian format |
| Reception times | PING the number of packets received during transmission | 2-byte, small endian format |
| Transmission times | PING the number of packets transmitted during transmission | 2-byte, small endian format |
| Reception time | The response time of the PING frame | 2-byte, small endian format |

Otherwise consistent with the control command frame parameters

### 6.5.2 Command Code

The higher 8 bits are the status of the current command code

| Bit | Function | Description |
|---|---|---|
| 7 | Answer flag | 0: Main control send command<br>1: Chip answer command |
| 6:4 | - | Reserved |
| 3:0 | Startup mode | 1: Statistical model<br>After the statistical mode is enabled, PING frames are sent according to the specified interval, and statistics are received. When the number of sending times reaches the specified interval, Ping frames are automatically terminated and the status is returned to the master.<br>2: Echo mode<br>When echo mode is enabled, each time after sending a PING frame, it counts the return time of the PING response and notifies the master control, which reads it and then starts the transmission again after the PING interval, and automatically ends and returns to the end status after reaching the set number of transmissions. |

The lower 8 bits are the status of the current command code

| Command code value | Function | Description |
|---|---|---|
| 0x00 | Test command | Used to test the sending and receiving of the PING channel command |
| 0x01 | Start PING | Start a PING task |
| 0x02 | Stop PING | Stop current PING task |
| 0x03 | Echo status | The chip automatically reports the PING status |
| 0x04 | End status | The chip automatically reports the PING task after |

| | | |
|---|---|---|
| | | the ping task is complete |
| 0x05 | Reset | Reset the PING channel status and stop the PING task |
| 0xE0 | Error status | The chip is reporting some error status |

① Test command: 0x00

This parameter is used to test the verification of PING channel sending and receiving and notifies the version information. All other fields are set to 0 after the flag, version, and command code are entered. The answer command is 0x8000. The master can verify the flag and version information.

The current version is V1.0.

Example: Main control send:

0x57,0xAB,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00

Chip response:

0x57,0xab,0x00,0x01,0x00,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00

② Start command: 0x01

Example 1: Main control send:

0x57,0xab,0x00,0x01,0xe8,0x03,0x90,0x01,0x01,0x00,0x28,0x00,0x05,0x00,0x01,0x02, 0xc0,0xa8,0x01,0x64

Interval: 1000mS

Timeout: 400mS

Data type: 0xff

Data length: 40

Transmission times: 5

Command code: Startup echo 0x0201

Ip: 192.168.1.100

Chip echo response:

0x57,0xab,0x00,0x01,0x03,0x80,0x01,0x00,0x01,0x00,0x01,0x00,0x00,0x00

Response status: Echo

Reception times: 1

Transmission times: 1

Response time: 0

Example 2: Main control send:

0x57,0xab,0x00,0x01,0xe8,0x03,0xf4,0x01,0x03,0x00,0x28,0x00,0x05,0x00,0x01,0x01, 0xc0,0xa8,0x01,0x64

Interval: 1000ms

Timeout: 500ms

Data type: Random

Data length: 40

Transmission times: 5

Command code: Wait for statistics 0x0101

Ip: 192.168.1.100

Chip response:

0x57,0xab,0x00,0x01,0x04,0x80,0x01,0x00,0x05,0x00,0x05,0x00,0x00,0x00
Answer status: ping is complete
ping status: ok
Reception times: 5
Transmission times: 5
Response time: 0

③ Stop command: 0x02

Example Stop the PING task. Parameter input version information, flag, and command code, other fields can be set to 0, the input parameter is automatically ignored by the chip, the response command is 0x8002, the response command frame contains the version information, flag, and command code.

Example: main control send
0x57,0xab,0x00,0x01,0xe8,0x03,0xf4,0x01,0x03,0x00,0x28,0x00,0x05,0x00,0x02,0x00,0xc0,0xa8,0x01,0x64
Interval: 1000ms
Timeout: 500ms
Data type: Random
Data length: 40
Transmission times: 5
Command code: Stop 0x0002
Ip: 192.168.1.100

Chip response:
0x57,0xab,0x00,0x01,0x02,0x80,0x01,0x00,0x02,0x00,0x02,0x00,0x00,0x00
Answer status: ping stop
ping status: ok
Reception times: 2
Transmission times: 2
Response time: 0

④ Echo status

The command output status is the CH392F active report command. The command output takes effect after the command output mode is enabled when the main control sends the start command. After the PING function is enabled, the command output is returned when the PING response is received or ICMP receive times out.
See startup command description for an example.

⑤ End command: 0x04

The end command is the CH392F active report command. When the number of times the start command is sent ends, the CH392F generates the end command and returns it to the master. And fill in the information with the corresponding results.
See startup command description for an example.

⑥ Reset command: 0x05

After the main control sends the reset command, the chip stops the PING task and clears all PING status. This command is used in some abnormal cases.

Example: Main control send:
0x57,0xab,0x00,0x01,0xe8,0x03,0xf4,0x01,0x03,0x00,0x28,0x00,0x05,0x00,0x05,0x00,0xc0,0xa8,0x01,0x64
Chip response:
0x57,0xab,0x00,0x01,0x05,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00

⑦ Error status report: 0xe0

The error status report is an active return command of the chip, which is used to report when an exception occurs during the execution of some commands or command operations. For example, after the PING task is stopped, the master sends a stop command code.

# 7. Parameters

## 7.1. Absolute Maximum Value

Critical value or exceeding the absolute maximum value may cause the chip to work abnormally or even be damaged.

| Name | Parameter description | | Min. | Max. | Unit |
|---|---|---|---|---|---|
| $T_A$ | Ambient temperature during operation | $V_{DD}$=3.3V | -40 | 85 | ℃ |
| $T_S$ | Ambient temperature during storage | CH392T | -40 | 125 | ℃ |
| | | CH392F | -40 | 105 | ℃ |
| $V_{DD}$ | Supply voltage (VDD connected to power supply, GND) | | -0.3 | 3.9 | V |
| $V_{IO}$ | Voltage on the input or output pins | | -0.3 | $V_{DD}$+0.3 | V |

## 7.2. Electrical Parameter

### 7.2.1 CH392T Electrical Parameter

| Name | Parameter description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $V_{DD}$ | System supply voltage | 3.0 | 3.3 | 3.6 | V |
| $V_{IL}$ | GPIO low input voltage | 0 | | 0.9 | V |
| $V_{IH}$ | GPIO high level input voltage | 2.0 | | $V_{DD}$ | V |
| $V_{OL}$ | Low level output voltage (8mA sink current) | 0 | | 0.4 | V |
| $V_{OH}$ | High level output voltage (8mA source current) | $V_{DD}$-0.4 | | $V_{DD}$ | V |
| $I_{idle}$ | Default running current of a chip after it is powered on | | 19.4 | | mA |
| $I_{init}$ | Chip running current after network initialization | | 23.7 | | mA |
| $I_{link}$ | Current after chip network connection | | 23.5 | | mA |
| $I_{send}$ | Current when the chip performs Ethernet transmission | | 40.0 | | mA |
| $I_{rec}$ | Current when the chip performs Ethernet reception | | 24.0 | | mA |
| $I_{sleep1}$ | Chip current in light sleep mode 1 | | 4.8 | | mA |
| $I_{sleep2}$ | Chip current in light sleep mode 2 | | 11.7 | | mA |
| $I_{sleep3}$ | Chip current in light sleep mode 3 | | 11.7 | | mA |
| $I_{stop}$ | Chip current in deep sleep | | 70.0 | | uA |

*Note: The current parameter is the average current value, and the test communication speed is about 2Mbps/S.*

### 7.2.2 CH392F Electrical Parameter

| Name | Parameter description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $V_{DD}$ | System supply voltage | 3.0 | 3.3 | 3.6 | V |

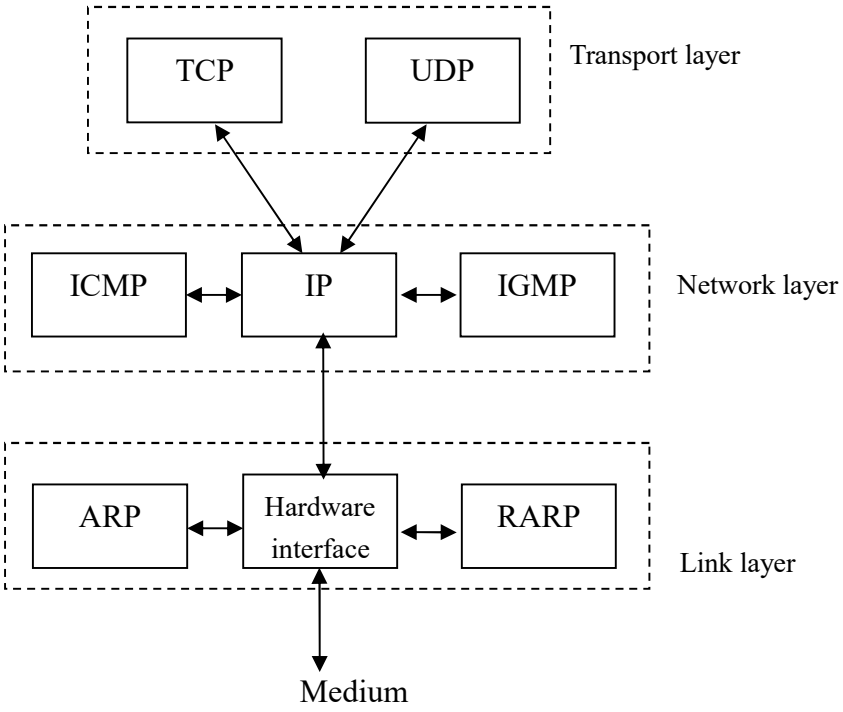
| | | | | | |
|---|---|---|---|---|---|
| $V_{IL}$ | GPIO low input voltage | 0 | | 0.9 | V |
| $V_{IH}$ | GPIO high level input voltage | 2.0 | | $V_{DD}$ | V |
| $V_{OL}$ | Low level output voltage (5mA sink current) | 0 | | 0.4 | V |
| $V_{OH}$ | High level output voltage (5mA source current) | $V_{DD}$-0.4 | | $V_{DD}$ | V |
| $I_{idle}$ | Default running current of a chip after it is powered on | | 4.8 | | mA |
| $I_{init}$ | Chip running current after network initialization | | 7.8 | | mA |
| $I_{link}$ | Current after chip network connection | | 7.5 | | mA |
| $I_{send}$ | Current when the chip performs Ethernet transmission | | 24.2 | | mA |
| $I_{rec}$ | Current when the chip performs Ethernet reception | | 7.7 | | mA |

*Note: The current parameter is the average current value, and the test communication speed is about 2Mbps/S.*

# 8. Application

## 8.1. Application Basis

CH392 internally integrates IPv4, ARP, ICMP, IGMP, UDP, TCP and other protocols, and their relationship diagram is as follows:




TCP and UDP are two important transport layer protocols, both of which use IP as the network layer protocol.

TCP is a connection-oriented transport that provides reliable byte stream transmission services.

UDP is a simple datagram-oriented transport layer protocol, unlike TCP, UDP does not guarantee that data packets accurately reach their destination.

TCP provides highly reliable communication for network devices. What it does is to divide the data handed to it by the application program into appropriate small chunks to the following network layer, confirm the received packets, set the timeout clock, etc. Because the transport layer provides highly reliable end-to-end communication, the application layer customers ignore all the details. UDP provides a very simple service for the application layer, faster than TCP, it just sends the datagram from one network terminal to another network terminal, but does not guarantee that the datagram can reach the other end, any necessary reliability must be provided by the application layer.

IP is the protocol on the network layer, which is simultaneously used by both TCP and UDP. Each set of data of TCP and UDP is transmitted in the network through the IP layer.

ICMP is an ancillary protocol of IP protocol, which is used by IP layer to exchange error messages or other important information with other hosts or routers, for example, when CH392 generates inaccessible interrupt, error messages are exchanged through ICMP. PING also uses the ICMP protocol.

IGMP is an Internet group management protocol, which is mainly used to multicast a UDP datagram to multiple hosts.

ARP is an address resolution protocol, which is used to translate the addresses used by the IP layer and the network interface layer.

For basic formats such as Ethernet frame, IP, UDP, TCP packet, etc., please refer to Section 8.2.

## 8.2. Application Reference Steps

This chapter introduces the common operation steps. Refer to the example program for details.

### 8.2.1. Initialize CH392 (necessary operation)

① Send the command CMD_SET_MAC_ADDR to set MAC address of CH392;

② Send the command CMD_SET_IP_ADDR to set IP address of CH392;

③ Send the command CMD_SET_GWIP_ADDR to set the gateway IP address of CH392;

④ Send the command CMD_SET_MASK_ADDR sets the subnet mask of CH392;

⑤ Send the command CMD_INIT_CH392 to initialize CH392;

⑥ After delay for more than 2mS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_INIT_CH392. If CH392_ERR_BUSY is returned, it will indicate that CH392 is internally executing the command and needs to execute ⑥ again; If CH392_ERR_SUCCESS is returned, it will indicate that the command is executed successfully. Generally, it takes 350mS to complete CMD_INIT_CH392.

Generally, the step ① is not required. MAC address assigned by IEEE has been burned when CH392 is delivered.

If DHCP is required to be started, the steps ②-④ will not be required.

The step ④ is optional. The default subnet mask is 255.255.255.0, which is generally not required to be set.

After CH392 receives the command CMD_INIT_CH392, it initializes the internal TCP/IP stack, MAC and PHY.

### 8.2.2. Initialize Socket to UDP Mode

Initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set Socket to work in UDP mode;

② Send the command CMD_SET_IP_ADDR_SN to set the destination IP address;

③ Send the command CMD_SET_DES_PORT_SN to set the destination port;

④ Send the command CMD_SET_SOUR_PORT_SN to set the source port;

⑤ Send the command CMD_OPEN_SOCKET_SN to turn on Socket;

⑥ After delay for more than 2mS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_OPEN_SOCKET_SN. If CH392_ERR_BUSY is returned, it will indicate that CH392 is internally executing the command and needs to execute ⑥ again. If CH392_ERR_SUCCESS is returned, it will indicate that the command is executed successfully; if other values are returned, it will indicate the failure to turn on Socket.

UDP message structure:

| Destination MAC | Source MAC | Type | IP header | UDP header | UDP data | CRC32 |
|---|---|---|---|---|---|---|
| 6 Byte | 6 Byte | 2 Byte | 20 Byte | 8 Byte | Max. 1472 Bytes | 4 Byte |

UDP is a simple, unreliable, datagram-oriented transport layer protocol with fast transmission speed, which cannot guarantee that the data can reach the destination. Therefore, the application layer must guarantee the reliable and stable transmission.

After the microcontroller writes several bytes of data stream to CH392, the CH392 data stream is encapsulated in

the UDP data section for sending. The maximum length of a UDP packet can be sent is 1472 bytes. If the data stream length of the single-chip microcomputer needs to be written is greater than 1472 bytes, multiple queues can be used to write, and the CH392 will encapsulate the data stream in the queue to send UDP packets. The length of bytes written by the single chip microcomputer shall not be greater than 1472 bytes, that is, the maximum UDP frame data, and the data can be written next time only after receiving the SINT_STAT_SENBUF_FREE interruption. If the main control needs to process and send the data quickly, it can query the queue cache status, and if the cache status is valid, it can execute the write command. In this case, the SINT_STAT_SENBUF_FREE interrupt is generated after each queue is sent.

When the CH392 receives a UDP packet, it copies the UDP data to the Socket receive buffer and generates a SINT_STAT_RECV interrupt. After receiving the interrupt, the MCU can send the command CMD_GET_RECV_LEN_SN to obtain the length of the data in the receive buffer. Then send the command CMD_READ_RECV_SN to read the data in the buffer. Since the CH392 cannot provide flow control in UDP mode, it is recommended that received data be read through quickly and in a timely manner to avoid data loss.

The CH392 supports two UDP modes: UDP client and UDP server. The UDP client can communicate with only the specified IP address and port, and the UDP server can communicate with any remote IP address and port.

There are some differences between the two in use:

① Initialization step ②: If the destination IP address is 0XFFFFFFFF, the Socket enters the UDP server mode; otherwise, the socket enters the UDP client mode.

② In client mode, CH392 directly sends the received data stream to the single chip computer. The single chip computer can read all the data at one time or only part of the data. In server mode, CH392 will add an 8-byte information table to the header of the data. The single-chip microcomputer can obtain the source information of the data packet according to the information table. The single-chip microcomputer must read all the data at one time.

| Data package length | Port | IP address | Data |
|---|---|---|---|
| 2 Byte | 2 Byte | 4 Byte | N Byte |

③ MCU sends data, and CH392 directly sends data to the destination IP and port specified during initialization in the client mode. In the server mode, CH392 can send data to any IP and port. MCU sets the destination IP and port before sending.

### 8.2.3. Initialize Socket to TCP Client Mode

Initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set Socket to work in TCP mode;

② Send the command CMD_SET_IP_ADDR_SN to set the destination IP address;

③ Send the command CMD_SET_DES_PORT_SN to set the destination port;

④ Send the command CMD_SET_SOUR_PORT_SN to set the source port;

⑤ Send the command CMD_OPEN_SOCKET_SN to turn on Socket;

⑥ After delay for more than 2mS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_OPEN_SOCKET_SN. If CH392_ERR_BUSY is returned, it will indicate that CH392 is internally executing the command and needs to execute ⑥again; If CH392_ERR_SUCCESS is returned, it will indicate that the command is executed successfully; if other values are returned, it will indicate the failure to turn on Socket;

⑦ Send the command CMD_TCP_CONNECT_SN to perform TCP connection;

⑧ After delay for more than 2mS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_TCP_CONNECT_SN. If CH392_ERR_BUSY is returned, it will indicate that CH392 is internally

executing the command and needs to execute ⑧ again; If CH392_ERR_SUCCESS is returned, it will indicate that the command is executed successfully. Other values indicate the failure to execute the command. Returning CH392_ERR_SUCCESS only indicates that the command is executed successfully and does not indicate a successful TCP connection. If TCP connection is successful, CH392 will generate the interrupt SINT_STAT_CONNECT. If the connection fails, CH392 will generate the interrupt SINT_STAT_TIM_OUT. If it is necessary to connect again, execute again from ⑤.

TCP message structure:

| Destination MAC | Source MAC | Type | IP header | TCP header | TCP data | CRC32 |
|---|---|---|---|---|---|---|
| 6 Byte | 6 Byte | 2 Byte | 20 Byte | 20 Byte | Max. 1460 Bytes | 4 Byte |

TCP provides a connection-oriented, reliable byte stream service.

The CH392 generates SINT_STAT_CONNECT, indicating that a TCP connection is established and data can be sent or received. Do not send data before the connection is established.

After the microcontroller writes several bytes of data stream to CH392, the CH392 data stream is encapsulated in the TCP data part for transmission. The maximum length of a TCP packet that can be sent is TCPMSS bytes. If the length of the data stream written by the microcontroller is greater than the TCPMSS bytes, CH392 will encapsulate the data stream into several TCP packets for sending. The length of bytes written by the microcontroller each time must not be greater than the length of the buffer of the send queue, and the next data write operation can be performed only after receiving the SINT_STAT_SENBUF_FREE interruption. If the master control needs to process the send quickly, it can query the cache status of the queue, and if the cache status is valid, it can execute the write command. In this case, the SINT_STAT_SENBUF_FREE interrupt is generated after each queue is sent. In TCP mode, if data fails to be sent, the SINT_STAT_TIM_OUT interrupt is generated, and the CH392 automatically disables the Socket.

When the CH392 receives a TCP packet, it copies the TCP data to the Socket receive buffer and generates a SINT_STAT_RECV interrupt. After receiving the interrupt, the MCU can send the command CMD_GET_RECV_LEN_SN to obtain the length of the data in the receive buffer. Then send the command CMD_READ_RECV_SN to read the data in the buffer. The single chip microcomputer can read all the data at one time or only part of the data. The remaining space in the receiving buffer is the TCP window. After each data read by the single chip microcomputer, CH392 will check the remaining space in the receiving buffer and inform the TCP server of the size of the current window. CH392F and CH392T have a slightly different sizing process for notifying the current window.

### 8.2.4. Initialize Socket to TCP Server Mode

Initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set Socket to work in TCP mode;

② Send the command CMD_SET_SOUR_PORT_SN to set the source port Sport;

③ Send the command CMD_OPEN_SOCKET_SN to turn on Socket;

④ Send the CMD_GET_CMD_STATUS command to obtain the CMD_OPEN_SOCKET_SN execution status after a delay of more than 2mS. If CH392_ERR_BUSY is displayed, the CH392 is executing commands internally. If the CH392_ERR_SUCCESS command is displayed, other values indicate that the Socket failed to be opened.

⑤ Send the CMD_TCP_LISTEN_SN command to start the TCP connection listening function. Then run the command to check whether the listening function is enabled.

In TCP server mode, if the client is connected and always be in the monitoring status on the Socket, the interrupt will not be generated. If TCP connection is successful, CH392 will generate the interrupt SINT_STAT_CONNECT. At this time, MCU can send the command CMD_GET_REMOT_IPP_SN to get the IP address and port number of the client.

The TCP server can connect to multiple TCP connections, and the microcontroller needs to set the source port of the Socket to be consistent with that of the server. If the TCP server listens to the connection, CH392 will check whether the source port of all the current sockets is consistent with that of the current server. The protocol type is TCP, and it is in the closed status. If found, this Socket is immediately opened, the connection is assigned to this Socket, and the MCU is notified that there is a connection event, if not found, the connection is reset. In this mode, the Socket of the server is only used for monitoring, and the MCU needs to allocate other sockets for the connection of the server. For example, if Socket0 is set to server mode and Socket1 and Socket2 are used to connect to this server, the steps are as follows:

Socket0 executes ①-④;

⑤ Send the command CMD_SET_SOUR_PORT_SN to Socket2 to set the source port Sport;

⑥ Send the command CMD_SET_PROTO_TYPE_SN to Socket1 to set Socket to work in TCP mode;

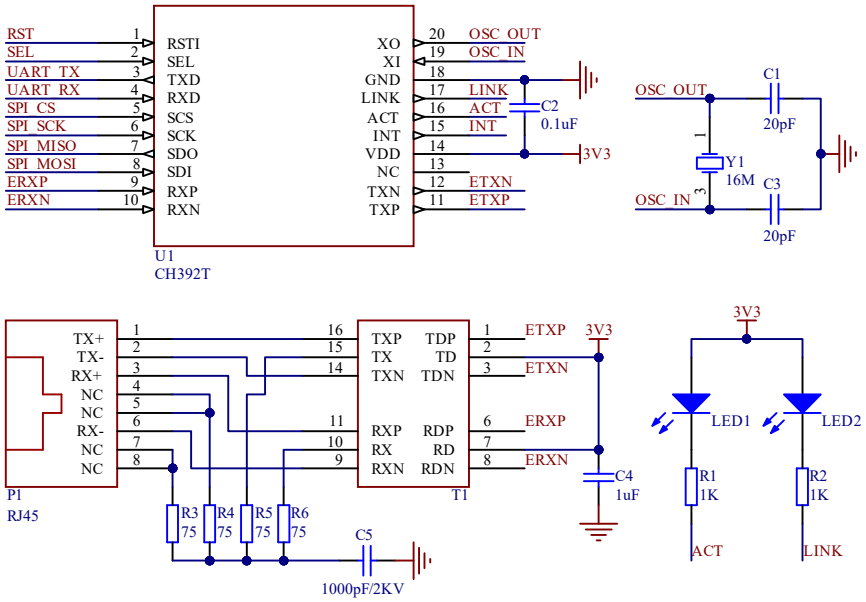
⑦ Send the command CMD_SET_SOUR_PORT_SN to Socket2 to set the source port Sport;

⑧ Send the command CMD_SET_PROTO_TYPE_SN to Socket2 to set Socket to work in TCP mode;

For data structures and the process for sending and receiving data, refer to the TCP client mode.

## 8.3. Hardware Circuit Design

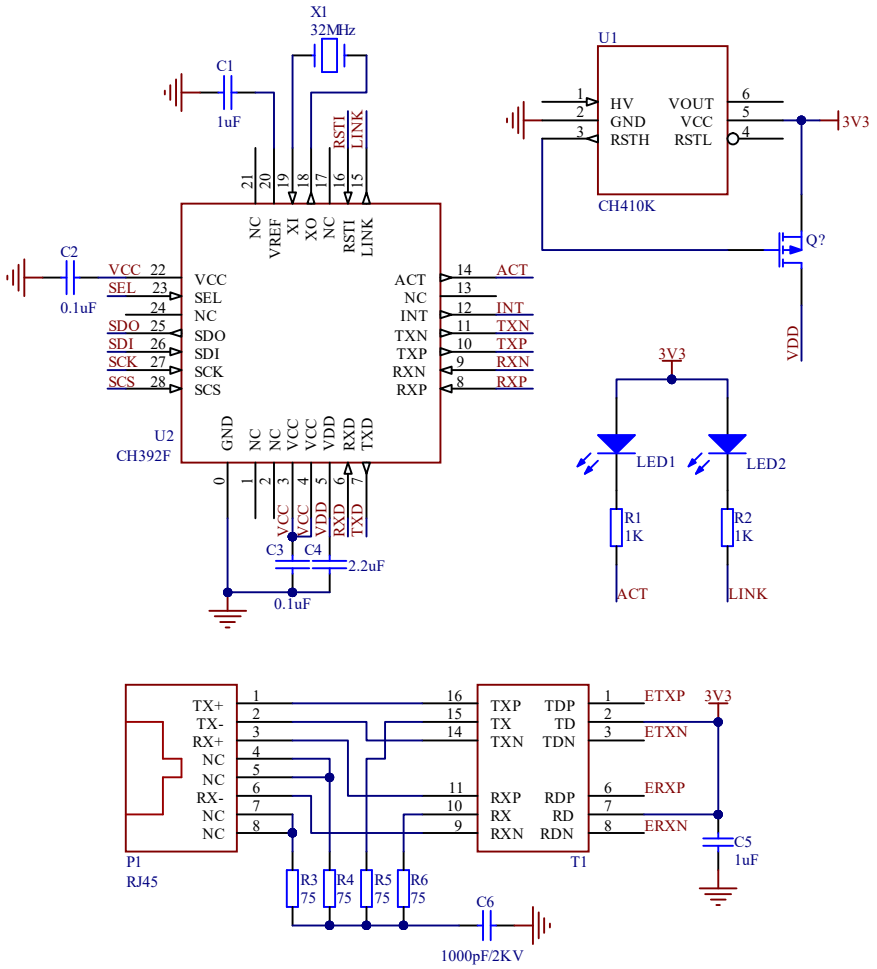
### 8.3.1. CH392T Hardware Circuit Design




Note: CH392T has built-in Ethernet 50Ω impedance matching resistor, do not connect 49.9Ω or 50Ω resistor externally.

CH392T needs external 8M crystal before version 0x18, and external 8M or 16MHz crystal is acceptable for version 0x18 and later.

### 8.3.2. CH392F Hardware Circuit Design




Note: CH392F has built-in Ethernet 50Ω impedance matching resistor, do not connect 49.9Ω or 50Ω resistor externally.

CH392F has built-in part of the oscillation capacitance of crystal X1, C7 and C8 can be adjusted according to the crystal parameters. For X1 with a load capacitance of 12pF, no crystal oscillator capacitor is needed; for X1 with a load capacitance of 20pF, it is recommended to connect a 15pF oscillator capacitor at each of its input and output terminals.